Introduction to Dynamic Programming Approach Using DP to solve the Fibonacci Numbers Problem

Week-06, Lecture-01

Course Code: CSE221

Course Title: Algorithms

Program: B.Sc. in CSE

Course Teacher: Masud Rabbani

Designation: Lecturer

Email: masud.cse@diu.edu.bd

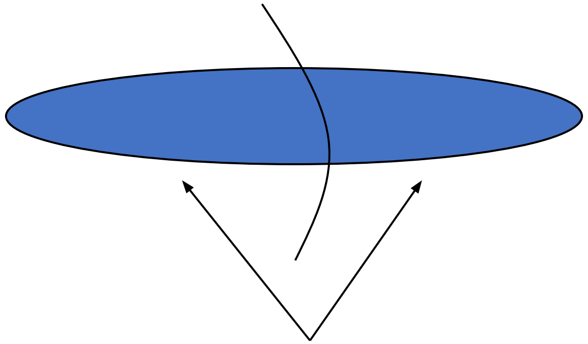
Dynamic Programming

- An algorithm design technique (like divide and conquer)
- Divide and conquer
  - Partition the problem into independent subproblems
  - Solve the subproblems recursively
  - Combine the solutions to solve the original problem

DP - Two key ingredients

- Two key ingredients for an optimization problem to be suitable for a dynamic-programming solution:

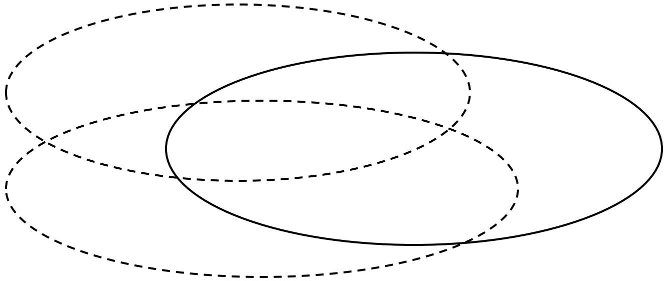
1. optimal substructures



Each substructure is optimal.

(Principle of optimality)

2. overlapping subproblems



Subproblems are dependent.

(otherwise, a divide-and-conquer approach is the choice.)

Three basic components

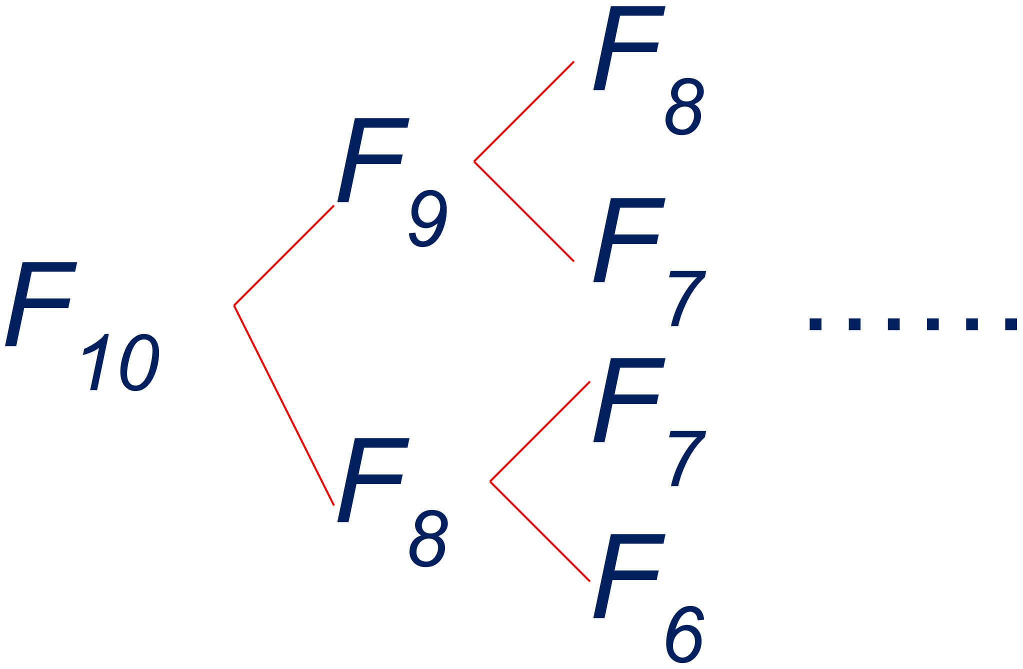
- The development of a dynamic-programming algorithm has three basic components:
    - The recurrence relation (for defining the value of an optimal solution);
    - The tabular computation (for computing the value of an optimal solution);
    - The traceback (for delivering an optimal solution).

Fibonacci numbers

The *Fibonacci numbers* are defined by the following recurrence:

$$F_0 = 0$$
$$F_1 = 1$$
$$F_i = F_{i-1} + F_{i-2} \quad \text{for } i > 1.$$

How to compute F_{10}?



F_{10}

F_9

F_8

F_8

F_7

F_7

F_6

……

Dynamic Programming

- Applicable when subproblems are not independent
  - Subproblems share subsubproblems

E.g.: Fibonacci numbers:

- Recurrence: F(n) = F(n-1) + F(n-2)
- Boundary conditions: F(1) = 0, F(2) = 1
- Compute: F(5) = 3, F(3) = 1, F(4) = 2

- A divide and conquer approach would repeatedly solve the common subproblems
- Dynamic programming solves every subproblem just once and stores the answer in a table

Tabular computation

- The tabular computation can avoid recompuation.

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
0	1	1	2	3	5	8	13	21	34	55

Result

Dynamic Programming Algorithm

1. Characterize the structure of an optimal solution
2. Recursively define the value of an optimal solution
3. Compute the value of an optimal solution in a bottom-up fashion
4. Construct an optimal solution from computed information

Textbooks & Web References

- Text Book (Chapter 15)
- Reference book iii (Chapter 19)
- www.codeforces.com
- www.topcoder.com

Thank you

&

Any question?